sages

Course code: **LEGACY**

Course title: **Working with Legacy Code**

Days: 3

Training partner: PRAGMATISTS

Description:

Course intended for:

The training is intended for programmers working with Legacy Code or a code expensive to maintain (in which making of small changes requires a lot of effort and/or results in many errors).

Course objective:

The training objective is to get the participants familiar with the threats and problems associated with working with a legacy code and the ways of overcoming these.

During the training, an algorithm of work with a legacy code is developed. The participants learn to work and make changes in the LC to minimize the risk of errors. The participants learn the refactoring techniques, allowing them to enhance the quality of a legacy code gradually.

Course strengths:

The training is conducted by trainers experienced in working with legacy codes. They lead the participants, step by step, through various aspects of working with LC.

As the training is visibly dominated by workshops, the participants are able to face unusual examples of use of LC (under the strict supervision of the trainer), acquire the practical skills in refactoring and testing of difficult codes.

Requirements:

The training participants are required to have the Java programming skills and be familiar with unit testing frameworks (e.g. JUnit, TestNG).

Course parameters:

3*8 hours (3*7 net hours) of lectures and workshops (with a visible emphasis on workshops).

Group size: no more than 8-10 participants.

Course curriculum:

1. Refactoring techniques
    - What is refactoring,
    - When (not to) refactor,
    - Code smells
    - Refactoring folder
    - Refactoring to patterns,
    - Use of IDE.
2. Legacy Code:
    - What is Legacy Code,
    - Risks associated with working with LC,
    - Why is it so difficult to work with
    - Methods of working with LC (Edit and Pray vs. Cover and Modify),
    - Algorithm of work with LC.
3. Seam Model:
    - What are seams,
    - How seams help in testing,
    - How to find seams.
4. Dependency Breaking techniques:
    - Which dependencies are the most problematic,

- How to break dependencies,
- A catalogue of dependency breaking techniques,
  - Hidden and global dependencies,
  - Huge methods,
  - Problematic dependencies to external libraries
  - Too many responsibilities.
- Effects of dependency breaking.

5. Effect Sketches:
   - Concluding on effects of changes made,
   - Finding the optimum testing points (interception points)
6. Characterization Tests
   - How to make changes to avoid making errors
   - How to select the context of characterization tests
7. Patterns of work with LC (Reengineering Patterns).